

# Final Report

## Android 2

### **Group 4**

Abraham Ackom-Mensah

Klaudia Konadu-Finke

Nikita Andrianov

Rahul Annadurai

# Table of content

Introduction	3
Product	3
Description	3
Functionality:	3
Potential functionalities	3
MOSCOW	3
Must have	3
Should have	3
Could have	3
GUI & Flow	4
Wireframes	4
Final GUI	6
Flow of application	12
Server	13
Collections	13
Storing and Retrieving data	13
Unit Test	14
Espresso Test	15

# Introduction

Throughout this document we will discuss and go through our application and its functionalities.

## Product

Name: Sportify

### Description

Sportify is an application which allows users to play sporting events with each other. It uses innovative android methods which allows users to create sporting events and find available sporting events near them.

### Functionality:

- Account creation.
- Search for sports near you.
- Location based search.
- Creation of sporting events.
- Participation of sporting events.
- Notification if new event has been added.

### Potential functionalities

- Directions to sport event through google maps.
- Leaving comments about events.
- Communication to sporting event participants.

## MOSCOW

### Must have

1. Users should be able to register, login and logout.
2. Support multiple user functionality.
3. User must be able to create an event.

### Should have

1. Show the location of an event and other events added by other people.
2. Users should be able to join an event.
3. A page where the user can view all his participation.

### Could have

1. Users can create/edit a profile page.
2. Filter list of events.
3. Comment section for events.
4. Update profile page.

## GUI & Flow

### Wireframes

Register

Register

Name

Email

Password

Password

Login

Login

Email

Password

Map

Distance  Category



Name : Name of event  
Location : Long & Lat  
Date : 26.11.2020  
Time : 12.00  
Proficiency : Beginner/Casual/Expert

Profile



Name : Firstname Lastname  
Email : email@mail.com  
Phone : 0161010101

## Single Event

Single Event screen layout:

- Header: Hamburger menu icon
- Form fields:
  - Name : Name of user
  - Location : Long & Lat
  - Date : 26. 11. 2020
  - Time : 0.00
  - Pricing : Beginner/Good/Expert
- Participants section:
  - Header: Participants
  - List of 4 items:
    - Person icon, Name : Name name
    - Person icon, Name : Name name
    - Person icon, Name : Name name
    - Person icon, Name : Name name
- Buttons: Leave, Join

## Participation

Participation screen layout:

- Header: Hamburger menu icon, + button
- List: Participations
  - Event Name, Date & Time
  - Event Name, Date & Time
  - Event Name, Date & Time
  - Event Name, Date & Time
  - Event Name, Date & Time
  - Event Name, Date & Time

## Create Event

Create Event screen layout:

- Header: Hamburger menu icon
- Form fields:
  - Name : [text input]
  - Time : [text input] : [text input]
  - Date : [Day] [Month] [Year]
  - Category : [dropdown menu]
  - Max People : [dropdown menu]
  - Location : [map icon]
- Button: Create

Edit Text

Button

Fragments

Drop down

Listview

# Final GUI

## Registration

Sportify

First name

Last name

Phone number

Email

Password

REGISTER

Already have an account? Sign in here!!!!.

## Login

Sportify

Email

Password

LOG IN

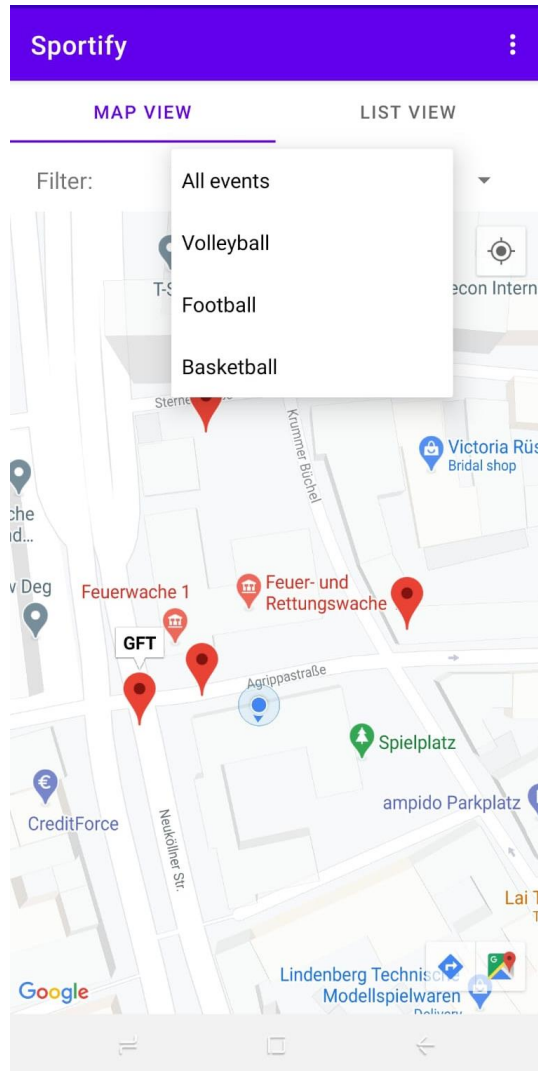
Don't have an account? Register here.

### Java and Xml Files

LoginActivity.java, activity\_login.xml, RegisterActivity.java, activity\_register.xml

The Login page is the default and first page on the app which you are taken to once the application is opened. This page makes use of a function called auth state listener, to know if you have logged in previously logged in before without logging out, if so, you are automatically logged in and redirected to the 'Home Page' once the application is opened, if not you are taken to the login in page where you can fill in your email and password.

## Event Map View

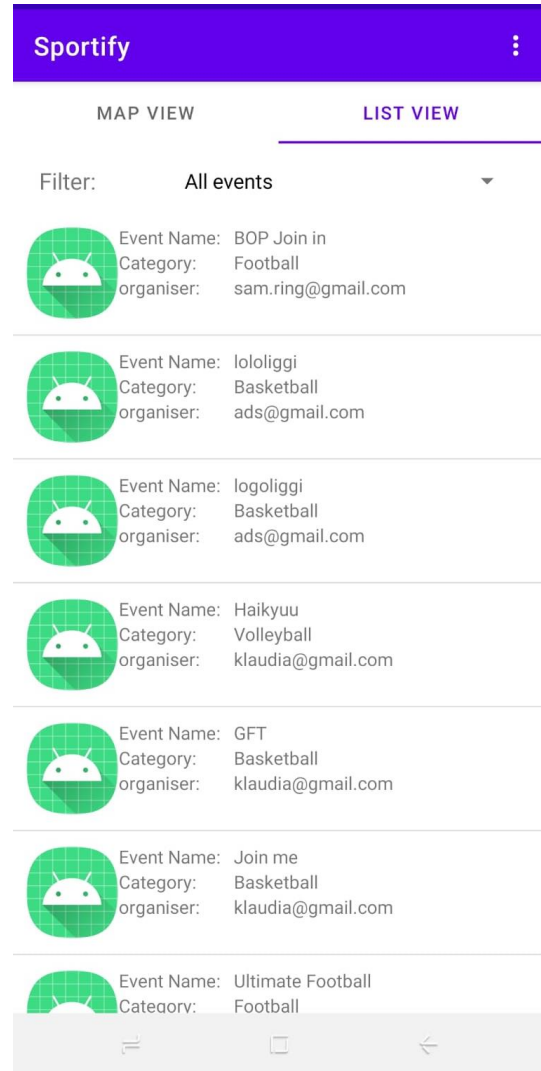


### Java and xml files:

MainActivity.java,  
HomePageEventsPagerAdapter.java  
MapFragment.java, fragment\_map.xml

This is the homepage, the first page after logging in, here the events are indicated with red pins, when clicking on one it will show the event's name. At the bottom right the direction symbol appears clicking on it will open Google Maps and directions will be given.

## Event List View



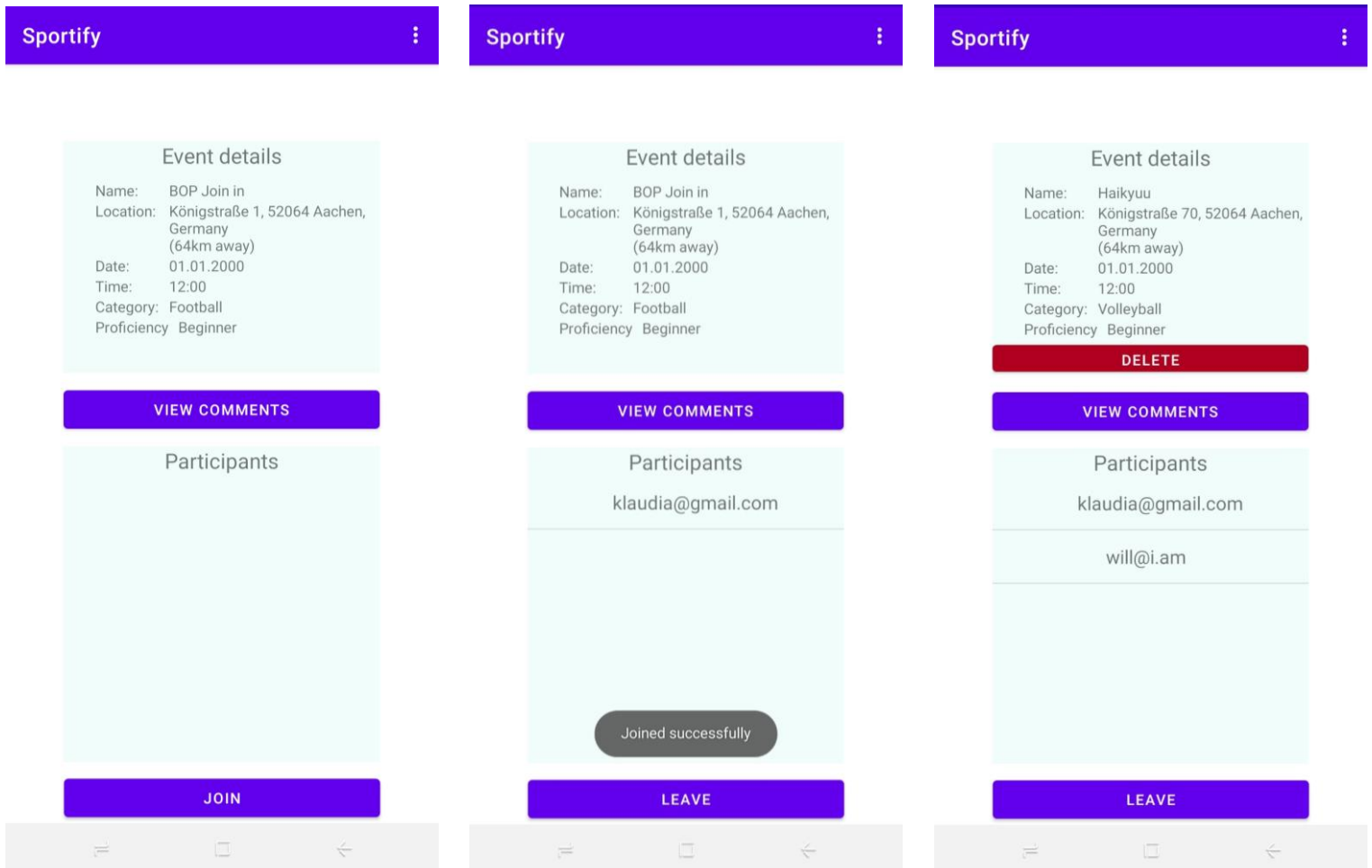
### Java and xml files:

MainActivity.java,  
HomePageEventsPagerAdapter.java

The events are also shown here in list format, with a bit more information, when clicking on an event the 'Single Event' page is opened.

The list and map can be filtered by categories using the dropdown at the top.

## Event page



### Java and xml files:

SingleEventPage.java , Participant.java, activity\_single\_event.xml, row\_participant.xml

The user will be taken to this page when an event is selected from 'Event list view' or 'Participations'. Here the user can view a list of participants, details, join/leave and leave a comment about the event. To join, he/she needs to click on the 'Join' button, if the user is already a participant, he/she can click on 'Leave' to be removed from the list.

Clicking on 'View comments' will open the 'Comment' page.

A user can delete an Event with the red button, only if he/she is the creator of the event.



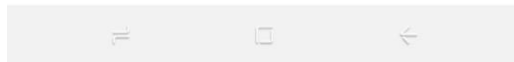
## Comment

**Sportify**

**POST**

klaudia@gmail.com

Event was great! Recommend



### Java and xml files:

Comment.java, CommentActivity.java,  
activity\_comments.xml, row\_comments.xml

Here the user can view comments others have left and post their own.

## Participations

**Sportify**

**Participations** **CREATE EVENT**

Haikyuu  
Volleyball  
Date n time

lololiggi  
Basketball  
Date n time

Catch me if you can!  
Basketball  
Date n time

BOP Join in  
Football  
Date n time

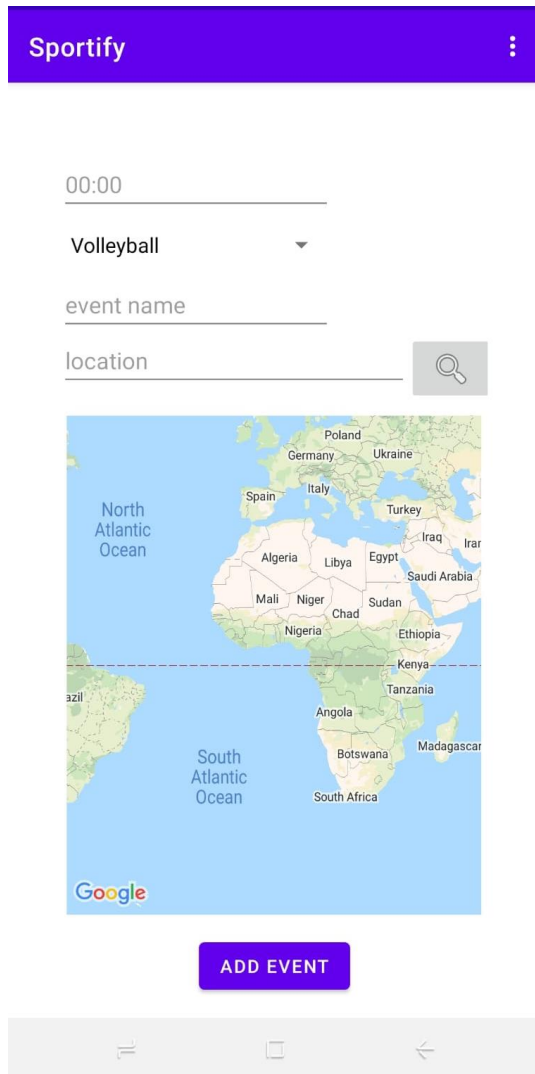


### Java and xml files:

Participation.java, ParticipationActivity.java,  
activity\_participation.xml, row\_participant.xml

On this page the user can view all the events that her/she has joined, clicking on an event will take them to the 'Single Event' page.  
If the 'Create Event' button is clicked they will be taken to the corresponding page.

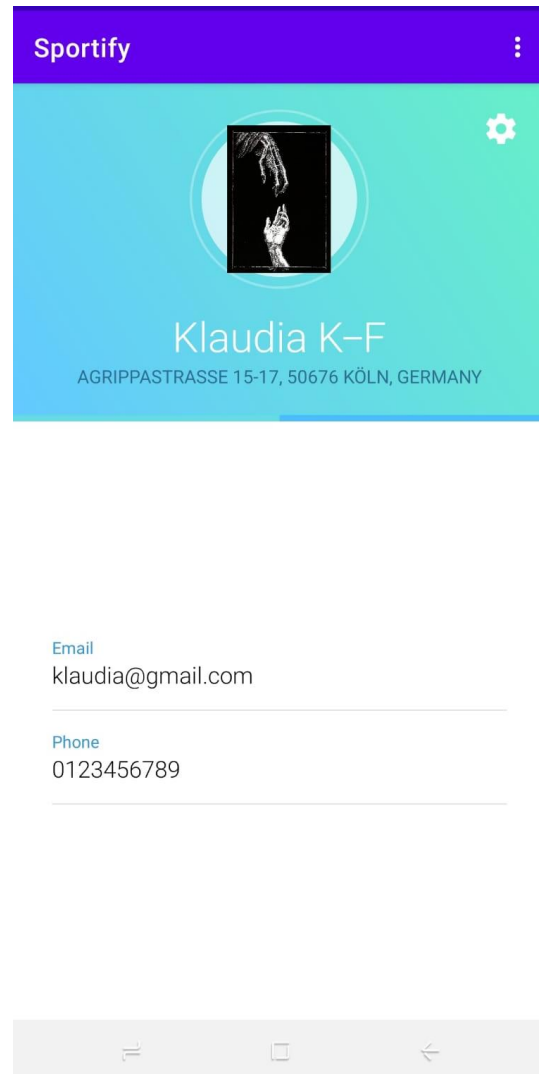
## Create Event



### Java and xml files:

CreateEventActivity.java, activity\_create\_event.xml  
On this page the user can create a new event by giving the sport category, event name and where it will take place.

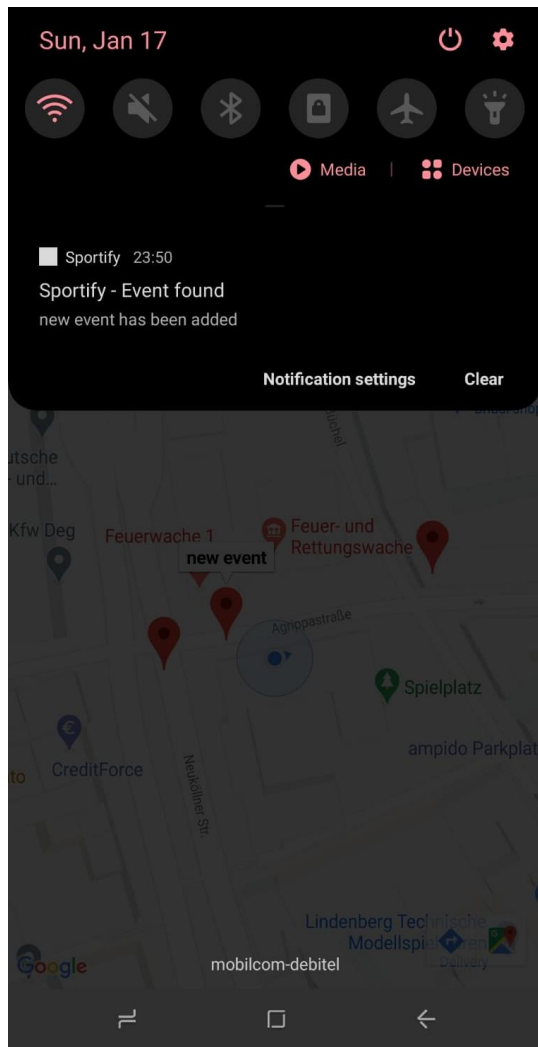
## Profile



### Java and xml files:

ProfileActivity.java, activity\_profile.xml  
Here the user can view the email they used to sign in as well as their phone number, apart from this his/her current location is shown and they can change their profile picture by clicking on the image.

# Notification

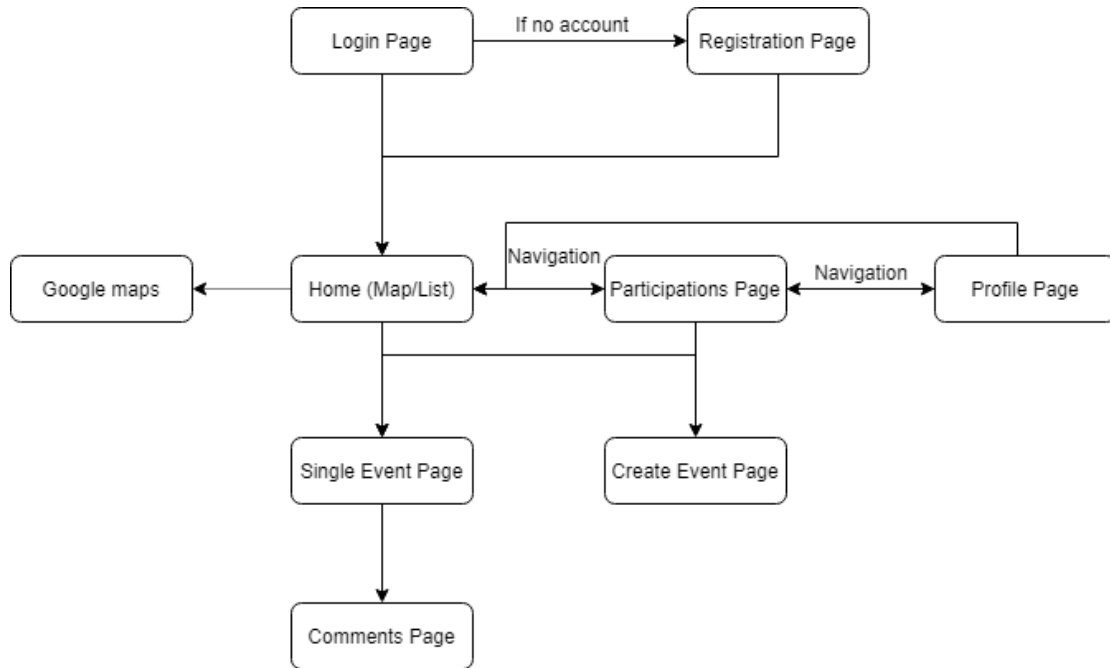


## Java and xml files

MapFragment.java, fragment\_map.xml

Once a new Event is created, and added to the database, anyone who is currently logged into the application will receive a notification showing them the name of the new event that has been added. Anyone who is not logged in once logged in, receives missed notifications.

## Flow of application



# Server

For this project we will be using the Firebase Realtime database. We chose this server as it syncs data every time a change is made therefore making it real-time and updates devices of other users.

Firebase also offers a great authentication feature making it easier to use and implement. Due to the pre-made API's the integration and set up was quick and simple.

## Collections

- Members
  - First name
  - Last name
  - Email
  - Password
  - Number
  - Profile
- Categories
  - Category name
  - Category id
- Events
  - Event id
  - Name
  - Category id
  - Location (longitude & latitude)
  - Creator
- Participants
  - Email
  - Event id
- Comments
  - Comment
  - Email
  - Event id

## Storing and Retrieving data

Data is being stored as a collection of documents using the JSON format. In the applications, Data is retrieved by attaching an asynchronous listener to a Firebase Database reference. The listener is triggered once for the initial state of the data and again anytime the data changes.

# Testing

## Unit Test

For unit testing both local and instrumented tests were conducted. The frameworks that were used for the unit tests are JUnit and Mockito.

Local Unit test was done for the Event Class.

```
✓ Tests passed: 1 of 1 test – 764 ms
"/Applications/Android Studio.app/Contents/jre/jdk/Contents/Home/bin/java" ...
Process finished with exit code 0
```

Local Unit test for Category Class

```
✓ Tests passed: 1 of 1 test – 12 ms
"/Applications/Android Studio.app/Contents/jre/jdk/Contents/Home/bin/java" ...
Process finished with exit code 0
```

Local Unit test for Participants Class was also successful.

```
✓ Tests passed: 1 of 1 test – 7 ms
"/Applications/Android Studio.app/Contents/jre/jdk/Contents/Home/bin/java" ...
Process finished with exit code 0
```

Instrumented test was done for LoginActivity

```
Testing started at 8:28 PM ...

01/19 20:28:16: Launching 'LoginActivityTest' on Pixel 2 Pie 9.0 - API 28.
App restart successful without requiring a re-install.
Running tests

$ adb shell am instrument -w -m -e debug false -e class 'com.example.sportify.LoginActivityTest' com.example.sportify.test/androidx.test.runner.
Connected to process 15216 on device 'pixel_2_pie_9_0_-_api_28 [emulator-5554]'.

Started running tests

Tests ran to completion.
```

## Espresso Test

To test User Interface, we have decided to use Espresso testing Framework, as it has simple workflow and fast feedback. During the development, the following activities have been tested:

- LoginActivity: The process of the login and if all components are being initiated on the activity load.
- MapActivity: If map and all components are being initiated on the activity load.

In order to test Sportify across a wide variety of devices, we have connected Android Studio with Firebase Test Lab - a cloud base test infrastructure.

